ELEC 474

Machine Vision

Lab 5

Appearance Based Facial Recognition (Eigenface)

Kevin Hughes

Due Date: Thursday, November 8th, 2012 in lab

In this lab you will implement Eigenface one of the original facial recognition techniques and a nearest neighbour algorithm.

Marking Scheme

Pre Lab	3 marks
Visualizing the Mean face	1 mark
Visualizing the Eigenfaces	2 marks
Nearest Neighbour Recognizer	3 marks
Overall Effort	1 mark
Total	10 marks

Part 1 - Combining Data Sets

Using the code from the pre-lab combine your data set with the data sets made by your classmates. Try and get data from at least 2 of your peers, note that there is an effort mark for this lab essentially for doing above the minimum.

Part 2 - Principal Component Analysis

To create the eigenspace we perform Principal Component Analysis of PCA on the face data. The samples matrix you have created has each face stored as a row vector so when using cv::PCA use the CV_DATA_AS_ROW flag. For the number of components you can enter 0 to keep all the components or 1 less than the number of input samples as the last principal component is redundant.

For quick reference here is a link to the OpenCV documentation page for PCA:
`http://docs.opencv.org/modules/core/doc/operations_on_arrays.html#pca`

Part 3 - Visualization

To gain some insight into how eigenface works you are going to visualize some of the matrices calculated by PCA. Some framework code for your visualizations has been setup in the supplied program however there is a few places where you need to fill in code. The first image you will create is the Mean face. The PCA object you initialized in the last part contains a mean image but it cannot simply be viewed as is using cv::imshow for 2 reasons. First it is still in the form of a row vector so you will need to reshape it into an image (hint you know the size the image is supposed to be). Secondly the PCA object has converted all the matrices to CV_32F and scaled them. To make the image viewable you will need to re-normalize it to be between 0 and 255 and convert it to the proper type (CV_8UC3). The correct way to do this procedure is to find the minimum and the maximum and scale this to be between 0 and 255. Place this code in the norm_0_to_255 function. Afterwards your mean face should look similar to this:

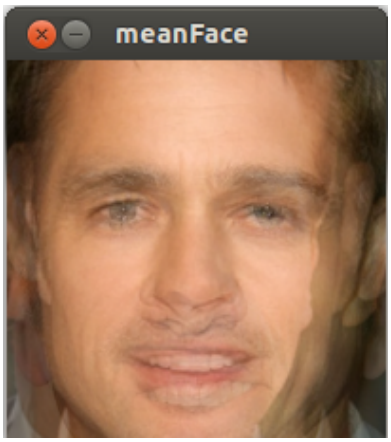

Figure 1: Mean Face

Next you will visualize the eigenfaces. The eigenfaces are simply the eigenvectors or principal components that were calculated by the PCA object. The eigenvectors are stored in rows just like the samples matrix you created. In the supplied lab program a loop has been provided that iterates through the eigenvectors, you will need to add code to reshape and normalize the eigenvectors just like the mean image so they can be viewed. After normalizing the eigenfaces try playing with OpenCV colormaps to improve your visualization. I used the JET color map and got this result:

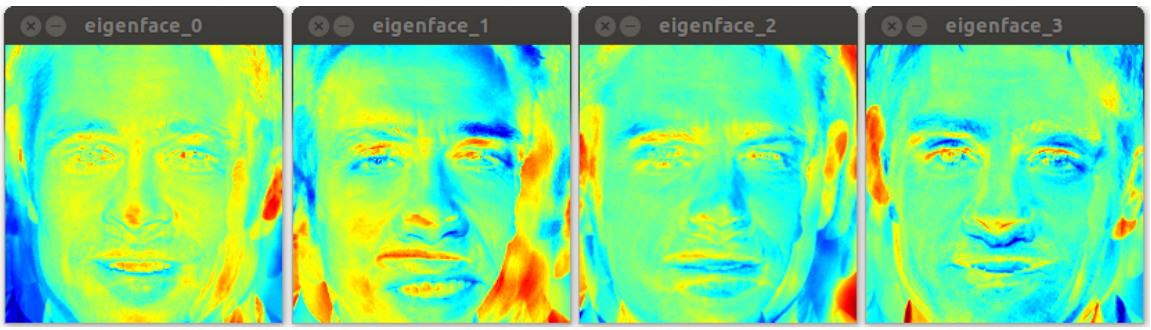

Figure 2: Eigenfaces

For reference:
`http://docs.opencv.org/modules/contrib/doc/facerec/colormaps.html`

Part 4 - Face Recognition

The first step to create the face recognition algorithm is to project all of our training data into the eigenspace. Fortunately we can do this all at once simply call the project method of OpenCV's PCA object with our sample matrix. It is easy to confirm that this has worked they way we want it to because if we print the matrix sizes after you will have the same number of rows before and after the projection only the number of columns will have decreased. Thanks to the dimensionality reduction performed by the PCA it is now feasible to compare new faces after they have been projected to the projected face database.

To finish your facial recognition algorithm fill in the *recognizeFace* function that will do just this. The algorithm you will implement to recognize a face is called the nearest neighbour algorithm which will check your query face against all the other faces and calculate the distance between them and then return the closest one as a match.

You can verify that your code works by trying each of the trained faces as the query face and checking that it returns the correct name. After this works find some new pictures of the people in your data set that are different than the trained ones, prepare this data and see if your algorithm correctly identifies the new faces.

As a final step in the lab take or get a picture of yourself and see who your face is closest too!